```
import java.util.PriorityQueue;

// Node class represents each character and its frequency
class Node implements Comparable<Node> {
    int freq;
    String symbol;
    Node left, right;
    String huff;

    Node(int freq, String symbol) {
        this.freq = freq;
        this.symbol = symbol;
        this.left = null;
        this.right = null;
        this.huff = "";
    }

    // Provide ordering based on frequency for PriorityQueue
    public int compareTo(Node o) {
        return this.freq - o.freq;
    }
}

public class HuffmanCoding {

    // Recursively print each character and their Huffman code
    public static void printNodes(Node node, String val) {
        if(node == null) return;

        String newval = val + node.huff;

        if(node.left == null && node.right == null) {
            System.out.println(node.symbol + " -> " + newval);
            return;
        }

        printNodes(node.left, newval);
        printNodes(node.right, newval);
    }

    public static void main(String[] args) {
        String[] chars = {"a", "b", "c", "d", "e", "f"};
        int[] freqs = {5, 9, 12, 13, 16, 45};

        PriorityQueue<Node> nodes = new PriorityQueue<>();

        // Add all nodes to priority queue
        for(int i = 0; i < chars.length; i++) {
            nodes.add(new Node(freqs[i], chars[i]));
        }

        // Build the Huffman Tree
        while(nodes.size() > 1) {
            Node left = nodes.poll();
            Node right = nodes.poll();

            left.huff = "0";
            right.huff = "1";

            Node newNode = new Node(left.freq + right.freq, left.symbol + right.symbol);
```

```
            newNode.left = left;
            newNode.right = right;

            nodes.add(newNode);
        }

        // Print codes
        printNodes(nodes.peek(), "");
    }
}
```

Input a, b, c, d, e, f with frequencies 5, 9, 12, 13, 16, 45

Output Huffman Codes:f → 0c → 100d → 101a → 1100b → 1101e → 111